IN THE CLAIMS:

Please find below a listing of all of the pending claims. The status of each claim is set forth in parentheses.

1. (Currently Amended) A compiler used by a computer architecture to compile a family of related functions, comprising:

a member recognizer configured to recognize a member function from said family of related functions, wherein each member function of the family of member functions is a mathematical function operable to be executed using a set of instructions and a portion of the set of instructions for each member function are identical;

a family start caller configured to make a family-start function call for said family of related functions, wherein the family-start function call is a call to a family-start function performing the identical set of instructions for each member function;

a member finish caller to make a member-finish function call for said member function, wherein the member-finish function call is a call to a member-finish function performing instructions unique to the member function; and

an optimizer configured to optimize said family-start function call;

wherein the optimized family-start function call causes execution of the portion of the set of instructions that are ~~common~~ identical for each member function ~~to the family of related functions~~ to occur prior to execution of instructions for each of a plurality of member-finish functions to reduce a number of instructions executed by the computer architecture in computing more than one member function from said family of related functions.

2. (Previously Presented) The compiler of claim 1, in which the optimizer is further configured to optimize said member finish function calls.

3. (Previously Presented) The compiler of claim 1, wherein said optimizer is configured to optimize on at least one of intermediate language level, architecture specific level, and operating system specific level.

4. (Previously Presented) The compiler of claim 1, wherein said optimizer is configured to in-line expand at least one of said family-start and member-finish calls.

5. (Previously Presented) The compiler of claim 1, wherein said optimizer includes common subexpression elimination, code motion, and dead-code elimination.

6. (Original) The compiler of claim 1, wherein said family of related functions includes at least one of trigonometric, hyperbolic, and square root functions.

7. (Original) The compiler of claim 1, wherein said family of related functions is identified by use of a data store.

8. (Original) The compiler of claim 7, wherein said data store includes at least one of a look-up table, an ascii file, a binary file, and a database file.

9. (Original) The compiler of claim 7, wherein said data store is modifiable.

10. (Original) The compiler of claim 1, wherein one or both of said family start caller and said member finish caller are configured to make said family-start and member-finish function calls, respectively, in an intermediate language.

11. (Original) The compiler of claim 10, wherein said intermediate language is non-architecture specific and non-operating system specific.

12. (Original) The compiler of claim 1, wherein said member-finish function call makes use of a result returned from said family-start function call.

13. (Currently Amended) A method to compile a family of related functions, comprising:

recognizing a member function from said family of related functions, wherein each member function of the family of member functions is a mathematical function operable to be executed using a set of instructions and a portion of the set of instructions for each member function are identical;

making a family-start call for said family of related functions, wherein the family-start function call is a call to a family-start function performing the identical set of instructions for each member function;

making a member-finish call for said member function, wherein the member-finish function call is a call to a member-finish function performing instructions unique to the member function; and

optimizing said family-start call to cause execution of the portion of the set of instructions that are ~~common~~ identical for each member function ~~to the family of related~~

~~functions~~ to reduce a number of instructions executed by a computer architecture in computing more than one member function from sail family of related functions.

14. (Previously Presented) The method of claim 13, further comprising:

optimizing member-finish function calls.

15. (Previously Presented) The method of claim 13 wherein said optimizing step includes: optimizing on at least one of intermediate language level and architecture specific level.

16. (Previously Presented) The method of claim 13 wherein said optimizing step includes: in-line expanding at least one of said family-start and member-finish calls.

17. (Previously Presented) The method of claim 13, wherein said optimizing step includes common subexpression elimination, code motion, and dead-code elimination.

18. (Original) The method of claim 13 wherein said family of related functions includes at least one of trigonometric, hyperbolic, and square root functions.

19. (Original) The method of claim 13 wherein said recognizing step includes: identifying said member function through a data store.

20. (Original) The method of claim 19 wherein said data store includes at least one of a look-up table, an ascii file, a binary file, or a database file.

21. (Original) The method of claim 19, further comprising:

modifying said data store.

22. (Original) The method of claim 13 wherein said family-start and member-finish function calls are made in an intermediate language.

23. (Original) The method of claim 22 wherein said intermediate language is non-architecture specific and non-operating system specific.

24. (Original) The method of claim 13 wherein said member-finish function call makes use of a result returned from said family-start function call.

25. (Previously Presented) The compiler of claim 1, wherein at least one calculation is almost identical for each member function of the family of related functions.

26. (Previously Presented) The compiler of claim 25, wherein at least one calculation is identical for each member function of the family of related functions.

27. (Previously Presented) The method of claim 13, wherein at least one calculation is almost identical for each member function of the family of related functions.

28. (Previously Presented) The method of claim 27, wherein at least one calculation is identical for each member function of the family of related functions.